Comput Optim Appl (2018) 71:331–352
https://doi.org/10.1007/s10589-018-0016-0

Mesh-based Nelder–Mead algorithm for inequality constrained optimization

Charles Audet[1] · **Christophe Tribes**[1]

Received: 9 November 2017 / Published online: 12 June 2018

Abstract Despite the lack of theoretical and practical convergence support, the Nelder–Mead (NM) algorithm is widely used to solve unconstrained optimization problems. It is a derivative-free algorithm, that attempts iteratively to replace the worst point of a simplex by a better one. The present paper proposes a way to extend the NM algorithm to inequality constrained optimization. This is done through a search step of the mesh adaptive direct search (Mads) algorithm, inspired by the NM algorithm. The proposed algorithm does not suffer from the NM lack of convergence, but instead inherits from the totality of the Mads convergence analysis. Numerical experiments show an important improvement in the quality of the solutions produced using this search step.

Keywords Nelder–Mead · MADS · Derivative-free optimization · Blackbox optimization · Constrained optimization

1 Introduction

The derivative-free Nelder–Mead algorithm (NM) was introduced in 1965 for unconstrained optimization problems in $\mathbb{R}^n$ [38]. Although the algorithm is very popular, it fails to solve some optimization problems [17,42]. In 1998, McKinnon [32] proposed

✉ Christophe Tribes
christophe.tribes@polymtl.ca

Charles Audet
Charles.Audet@gerad.ca

1 GERAD and Département de Mathématiques et Génie Industriel, École Polytechnique de Montréal, C.P. 6079, Succ. Centre-ville, Montreal, QC H3C 3A7, Canada

a smooth strictly convex function in $\mathbb{R}^2$ on which the sequence of trial points produced by the NM algorithm fails to approach the unique minimizer.

Different variants using sufficient decrease in the objective function value were proposed [26,37,45] to guarantee convergence to a critical point in the smooth case.

Price et al. [40] propose a frame-based NM method [15] requiring sufficient decrease. Gao and Han [20] study the behavior of the NM algorithm on a convex objective function, and propose an efficient variant for larger problems with adaptive simplex parameters.

A stochastic version [11] is presented in the nonsmooth case. Brea [9] presents a NM inspired algorithm for nonlinear mixed problems. Bűrmen et al. [10] adapt NM so that the trial points are generated on an underlying mesh, i.e., a discretization of the space of variables. The present paper pushes further in that direction, but for inequality constrained optimization without derivatives. Specifically, the NM algorithm is inserted in the search step of the *mesh adaptive direct search* (Mads) algorithm [2,3]. The resulting algorithm named Mads-NM benefits from the same hierarchical convergence analysis for nonsmooth constrained optimization as Mads.

The paper is structured as follows. Section 2 presents the original NM algorithm for unconstrained optimization. Section 3 gives a high-level description of the Mads algorithm, but focuses only on the elements necessary for the present work. Next, Sect. 4 proposes a way to insert a NM search step within Mads. Section 5 reports numerical experiments. Intensive tests are conducted on a collection of 87 test problems to tune the new algorithmic parameters. The resulting algorithm is then launched without any additional parameters, and is applied to three constrained blackbox engineering test problems. Our numerical experiments suggest that inserting a NM search step into the Mads algorithm significantly improves its performance.

2 The NM unconstrained optimization algorithm

There are many ways to present the NM algorithm for unconstrained optimization

$$\min_{x \in \mathbb{R}^n} f(x)$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ is the objective function. We present it in a way that might seem overly complicated, but it allows to simplify the presentation of the inequality-constrained case. The following definitions are introduced to order trial points.

Definition 2.1 A point $x \in \mathbb{R}^n$ is said to *dominate* $y \in \mathbb{R}^n$, denoted $x \prec y$, if $f(x) < f(y)$.

In the situation were two trial points share the same objective function value, (i.e., $f(x) = f(y)$), then instead of ordering them arbitrarily, we resort to the following determinist function that returns the oldest of the points.

Definition 2.2 A point $x \in \mathbb{R}^n$ is said to be *older* than $y \in \mathbb{R}^n$, if x was generated by the algorithm before y. The function $\mathsf{Older} : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}^n$

$$\mathsf{Older}(x, y) = \begin{cases} x & \text{if } x \text{ was generated by the algorithm before } y, \\ y & \text{otherwise.} \end{cases}$$

returns the *oldest* of two points.

Together with the definition of dominance, the function Older allows us to introduce the following rule to compare two trial points.

Definition 2.3 The function $\mathsf{Best} : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}^n$

$$\mathsf{Best}(x, y) = \begin{cases} x & \text{if } x \prec y \\ y & \text{if } y \prec x \\ \mathsf{Older}(x, y) & \text{if } f(x) = f(y) \end{cases}$$

returns the *best* of two points x and y in $\mathbb{R}^n$. A point $x \in \mathbb{R}^n$ is said to be *better* than $y \in \mathbb{R}^n$, if $x = \mathsf{Best}(x, y)$.

The function Best is clearly transitive, commutative and $\mathsf{Best}(x, x)$ is well-defined as it returns x.

The NM algorithm can now be presented using the above definitions. Each iteration of the NM algorithm starts with a set $\mathbb{Y} = \{y^0, y^1, \dots, y^n\}$ of $n+1$ affinely independent points in $\mathbb{R}^n$ which defines a simplex. The simplex points are ordered so that y^{i-1} is better than y^i for every $i = 1, 2, \dots, n$. The objective of each NM iteration is to replace the worst simplex point y^n by a better one. In order to achieve this, NM uses some of the following points and parameters:

$x^c = \frac{1}{n}\sum_{i=0}^{n-1} y^i$	the centroid of the n best points;
$x^r = x^c + (x^c - y^n)$	the reflection of the worst point with respect to the centroid;
$x^e = x^c + \delta^e(x^c - y^n)$	where $\delta^e \in]1, \infty[$ is the expansion parameter;
$x^{oc} = x^c + \delta^{oc}(x^c - y^n)$	where $\delta^{oc} \in]0, 1[$ is the outside contraction parameter;
$x^{ic} = x^c + \delta^{ic}(x^c - y^n)$	where $\delta^{ic} \in] - 1, 0[$ is the inside contraction parameter;
$\gamma \in]0, 1[$	is the shrink parameter.

At each iteration, NM either replaces the last point y^n of the ordered simplex by either x^r, x^e, x^{oc} or x^{ic}, or it shrinks the simplex by replacing every vertex but the best one y^0, so that the simplex becomes

$$\mathbb{Y} \leftarrow \{y^0, y^0 + \gamma(y^1 - y^0), y^0 + \gamma(y^2 - y^0), \dots, y^0 + \gamma(y^n - y^0)\}. \quad (1)$$

The most widely used values of the parameters are

$$\delta^e = 2, \quad \delta^{ic} = -\frac{1}{2}, \quad \delta^{oc} = \frac{1}{2} \quad \text{and} \quad \gamma = \frac{1}{2}.$$

The comparisons to determine the new simplex are made relatively to the following zones.

Definition 2.4 (**Zones for unconstrained optimization**) Let $\mathbb{Y} = \{y^0, y^1, \ldots, y^n\}$ be an ordered simplex in $\mathbb{R}^n$. The trial point $x \in \mathbb{R}^n$

- Belongs to the inside contraction zone if $y^n \prec x$;
- Else, it belongs to the expansion zone if $x \prec y^0$;
- Else, it belongs to the reflection zone if x dominates at least 2 points of $\mathbb{Y}$;
- Else, it belongs to the outside contraction zone and x dominates 0 or 1 point of $\mathbb{Y}$.

The NM algorithm is compactly written in Algorithm 1.

Algorithm 1: The Nelder–Mead unconstrained algorithm (NM)

```
Given the vertices of an initial simplex $\mathbb{Y} = \{y^0, y^1, \ldots, y^n\}$.
1. Update the simplex:
    Reorder $\mathbb{Y}$.
2. Build a new simplex:
    If $x^r$ belongs to the inside contraction zone
        if $x^{ic}$ belongs to the inside contraction zone, shrink $\mathbb{Y}$ using Equation (1),
        otherwise set $y^n \leftarrow x^{ic}$;
    else if $x^r$ belongs to the expansion zone, then set $y^n \leftarrow \mathsf{Best}(x^r, x^e)$;
    else if $x^r$ belongs to the reflection zone, then set $y^n \leftarrow x^r$;
    else ($x^r$ belongs to the outside contraction zone), set $y^n \leftarrow \mathsf{Best}(x^r, x^{oc})$.

    Go to 1.
```

Figure 1 provides a graphical interpretation of the mechanism of the NM algorithm. The vertical axis represents the objective function f, and the six circles represent the objective function values $f(y^0) < f(y^1) < \cdots < f(y^5) = f(y^n)$. Replacing the vertex y^n by x^{oc} can only occur when x^r belongs to the outside contraction zone depicted in the figure, replacing y^n by x^{ic} can only occur when x^r belongs to the inside contraction zone, etc.

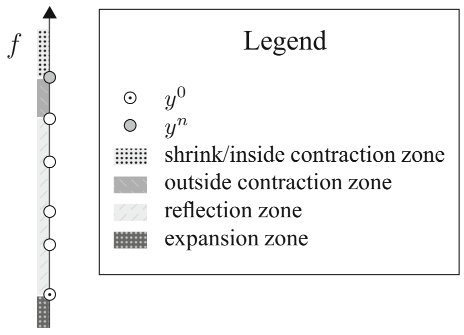


Fig. 1 Zones relative to a 6-point simplex

3 The Mads constrained optimization algorithm

We consider the Mads algorithm with the progressive barrier (PB) [3] to handle inequality constraints and with dynamic scaling [8] to handle the varying magnitudes of the variables. The target class of optimization problems is

$$\min_{x \in X \subset \mathbb{R}^n} \quad f(x)$$
$$\text{s.t.} \quad c(x) \leq 0,$$

where $f : X \mapsto \mathbb{R} \cup \{\infty\}$ and $c : X \mapsto (\mathbb{R} \cup \{\infty\})^m$ are functions with $c = (c_1, c_2, \ldots, c_m)^\top$, and X is some subset of $\mathbb{R}^n$. Notice that the functions are allowed to return the value ∞, which is a convenient technique to represent evaluation failures. The entire feasible region is denoted by the set $\Omega = \{x \in X : c(x) \leq 0\}$. The Mads algorithm targets problems on which no assumptions on the smoothness of f or c are made [5].

The PB uses the constraint violation function [18,19]

$$h(x) := \begin{cases} \sum_{j \in J} \left(\max\{c_j(x), 0\}\right)^2 & \text{if } x \in X, \\ \infty & \text{otherwise,} \end{cases}$$

where $J = \{1, 2, \ldots, m\}$ is the set of indices of the constraints. The constraint violation function value $h(x)$ is nonnegative, and equals zero if and only if the point x belongs to Ω.

Mads is a search-poll direct search iterative algorithm in which each iteration is constituted of two main steps. The SEARCH step can use various strategies (including utilization of surrogate functions and heuristics) to explore the space of variables. The POLL step follows stricter rules and performs a local exploration in the space of variables in a region delimited by the *poll size vector* $\Delta^k \in \mathbb{R}^n_+$. In practice, a well conceived SEARCH step can allow the algorithm to escape locally optimal solutions (e.g., [1,7]), and the POLL step ensures practical and theoretical convergence [2].

Both these steps generate their candidate points on a discretization of the space of variables called the *mesh*. The fineness of the discretization is controlled by the *mesh size vector* $\delta^k \in \mathbb{R}^n_+$. Formally, at iteration k, the mesh is defined as follows:

$$M^k = V^k + \left\{\operatorname{diag}(\delta^k) z : z \in \mathbb{Z}^n\right\}. \tag{2}$$

where *the cache* V^k is the set of points that were visited by the algorithm by the start of iteration k. In other words, a point x belongs to V^k if and only if both $f(x)$ and $h(x)$ were evaluated by the start of iteration k. The first set V^0 is the set of one or more initial points supplied by the user.

The mesh and poll size vectors are updated at the end of each iteration. The entries of both vectors are reduced when the iteration fails to improve the incumbent solution, and otherwise, they are either increased or remain at the same value. Algorithm 2 provides a high-level description of Mads; the reader is invited to consult [8] for a thorough description.

Algorithm 2: The mesh adaptive direct search algorithm (Mads)

```
Given a user-defined set of starting point: $V^0 \subset \mathbb{R}^n$,
and initial mesh and poll size vectors: typically $\delta_i^0 = \Delta_i^0 = 1$ for $i = 1, 2, \dots, n$.
Set the iteration counter: $k \leftarrow 0$.
1. Search step (optional):
    Launch the simulation on a finite set $S^k$ of mesh points.
    If successful, go to 3.
2. Poll step:
    Launch the simulation on the set $P^k$ of poll points.
3. Updates:
    Update the cache $V^{k+1}$, the incumbent $x^{k+1}$
    and the mesh and poll size vectors $\delta^{k+1}$ and $\Delta^{k+1}$.
    Increase the iteration counter $k \leftarrow k + 1$ and go to 1.
```

The fundamental convergence result [3] of the Mads algorithm states that if the entire sequence of trial points belongs to a bounded set, and if the set of so-called refining directions is sufficiently rich, then there exists an accumulation point x^* such that the generalized directional derivative $f^\circ(x^*; d)$ of Clarke [13] is nonnegative in every hypertangent [24] direction d to the domain Ω at x^* provided that x^* is feasible. A similar result (involving h rather than f) holds in situations where the iterates never approach the feasible region: if the accumulation point x^* is infeasible, then the generalized directional derivative $h^\circ(x^*; d)$ is nonnegative in every hypertangent direction d to the set X at x^*.

These convergence results hold regardless of the search step of Algorithm 2, as long as the set S^k is finite and belongs to the mesh M^k. In the present work, we make sure that these conditions are satisfied, which allows the convergence analysis to be unaltered.

4 The Mads-NM constrained optimization algorithm

We now present the NM-inspired Mads search step. Section 4.1 extends the definitions of the dominance relation $\prec$ and of the function Best to the constrained case. Section 4.2 focuses on the main Mads elements that need to be adjusted in order to implement the NM search feature. Section 4.3 describes the algorithmic specifications of the NM search step.

The design of this new algorithm involves a total of three parameters, whose default values are tuned in Sect. 5. These parameters are denoted by $\pi = (\pi_{\text{radius}}, \pi_{\text{svd}}, \pi_{\text{eval}})$. They refer to a sampling radius, a threshold on singular values and a threshold on the number of function evaluations, respectively.

4.1 Ordering trial points

The NM algorithm for unconstrained optimization hinges on the notion of ordering vertices from best to worst, and at each iteration, the algorithm attempts to replace the worst one. Ordering the points in a simplex is easy in the absence of constraints, as only the objective function values need to be compared. Ordering gets more technical in constrained optimization. In order to define a transitive relation between the trial points, the definition of dominance as well as the function returning the best of two points are adapted. The definitions rely on both the objective and constraint violation functions f and h.

Definition 4.1 A point $x \in \mathbb{R}^n$ is said to *dominate* $y \in \mathbb{R}^n$, denoted $x \prec y$, if

- Both points are feasible and $f(x) < f(y)$; or
- Both points are infeasible and $f(x) \leq f(y)$ and $h(x) \leq h(y)$ with at least one strict inequality.

The above definition is coherent with the terminology used with the progressive barrier [3], in which the feasible and infeasible points were treated differentely.

In the situation where two trial points share the same function values, i.e., $f(x) = f(y)$ and $h(x) = h(y)$, then instead of ordering them arbitrarily, we resort once again to the determinist function Older from Definition 2.2 returning the oldest of two points. Together with the definition of dominance, the function Older allows us to introduce the following rule to compare two trial points x and y.

Definition 4.2 The function Best : $\mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}^n$

$$\mathsf{Best}(x, y) = \begin{cases} x & \text{if } x \prec y \text{ or if } h(x) < h(y) \\ y & \text{if } y \prec x \text{ or if } h(y) < h(x) \\ \mathsf{Older}(x, y) & \text{if } f(x) = f(y) \text{ and } h(x) = h(y) \end{cases}$$

returns the *best* of two points x and y in $\mathbb{R}^n$. A point $x \in \mathbb{R}^n$ is said to be *better* than $y \in \mathbb{R}^n$, if $x = \mathsf{Best}(x, y)$.

Observe that with this definition, the comparison of a feasible point x with an infeasible one y always yields $x = \mathsf{Best}(x, y)$. The comparison of two infeasible points where none dominates the other returns the most feasible one in terms of h.

Proposition 4.3 *The function* Best *is transitive over a set of trial points.*

Proof Consider the three trial points x, y and z in $\mathbb{R}^n$ such that $x = \mathsf{Best}(x, y)$ and $y = \mathsf{Best}(y, z)$. We need to show that $x = \mathsf{Best}(x, z)$. The analysis is divided into four cases.

- First, if x and z are both feasible, then y is also feasible and therefore $x = \mathsf{Best}(x, z)$.
- Second, if x and z are both infeasible, then it follows that $0 < h(x) \leq h(y) \leq h(z)$. Therefore, the only way that $z = \mathsf{Best}(x, z)$ would be that

$$f(x) = f(y) = f(z), \quad h(x) = h(y) = h(z) \quad \text{and } z = \mathsf{Older}(x, z).$$

But if this were the case, then this would contradict $x = \mathsf{Older}(x, y)$ and $y = \mathsf{Older}(y, z)$.

- Third, if x is feasible and z is infeasible, then the observation preceeding the statement of the proposition ensures that $x = \mathsf{Best}(x, z)$.
- Finally, the case where x is infeasible and z is feasible is impossible, because if it were the case, then y would not be feasible since $x = \mathsf{Best}(x, y)$ and y would not be infeasible since $y = \mathsf{Best}(y, z)$. □

4.2 Construction of a simplex on a mesh

Section 4.3 defines a search step of Mads inspired by the NM algorithm. This NM search step uses elements present in the quadratic model search step proposed in [14]. In that paper, quadratic models of the objective and of each of the constraint functions are built and used to identify candidate trial points. But recall that an important requirement of the Mads algorithm is that every candidate point belongs to the mesh M^k. The following notation is adopted. Let x be some point in $\mathbb{R}^n$. By appending the subscript $\oplus$, we denote $x_\oplus \in M^k$ as the mesh point which is the closest to x (ties are broken by rounding up), as detailed in [6]. We will say that $x_\oplus$ is the point x rounded to the mesh. The quadratic models of [14] are constructed by considering all previously visited trial points within regions centred around incumbent solutions and of size related to Δ^k. A similar region is considered in the present work.

With the progressive barrier algorithm [3], there are up to two incumbent solutions called the *primary* and *secondary* poll centers. One of them is the best feasible solution found so far by the algorithm, and the other one is the infeasible solution with the smallest value of h. For constructing the simplex, only the primary poll center is considered. Given the incumbent solution x^k, the poll size vector Δ^k and the cache V^k, define the set

$$\mathbb{T}_{\pi_{\mathsf{radius}}} = \left\{ x \in V^k : |x_i - x_i^k| \le \pi_{\mathsf{radius}} \Delta_i^k, \; i \in \{1, 2, \ldots, n\} \right\},$$

where $\pi_{\mathsf{radius}} \ge 1$ is a fixed parameter called the sampling radius (it is the first of the three new algorithmic parameters). As in Algorithm 1, the points of $\mathbb{T}_{\pi_{\mathsf{radius}}}$ are reordered so that $x^{i-1} = \mathsf{Best}(x^{i-1}, x^i)$ for $i = 1, 2, \ldots, p$.

Algorithm 3 uses the set $\mathbb{T}_{\pi_{\mathsf{radius}}}$ to attempt the iterative construction of a simplex $\mathbb{Y}$. It also uses π_{svd}, the second of the three new algorithmic parameters. The algorithm considers points of $\mathbb{T}_{\pi_{\mathsf{radius}}}$ in sequence, initializes $\mathbb{Y}$ so that it contains only x^0 and then goes through the following steps before adding a point x^i to $\mathbb{Y}$. First, the matrix A formed by the columns of $\{y - x^0 : y \in \mathbb{Y}, y \neq x^0\}$ together with $\{x^i - x^0\}$ is constructed. Second, in order to ensure that the final simplex is far from being degenerate relative to the poll size vector, it is required that each singular value of the matrix $\mathrm{diag}(\Delta^k)^{-1} A$ exceeds a given threshold π_{svd}.

Algorithm 3: Construction of an ordered simplex $\mathbb{Y}$

```
Given the parameters $\pi_{\text{radius}} \geq 1$, $\pi_{\text{svd}} > 0$ and the set $\mathbb{T}_{\pi_{\text{radius}}} = \{x^0, x^1, \ldots, x^p\}$.
1. Construction of the simplex :
    Reorder $\mathbb{T}_{\pi_{\text{radius}}}$ and set $\mathbb{Y} = \{x^0\}$.
    For $i = 1$ to $p$ do
        let $A$ be formed by the columns of $\{y - x^0 : y \in \mathbb{Y}, y \neq x^0\} \cup \{x^i - x^0\}$,
        if all singular values of $\text{diag}(\Delta^k)^{-1} A \geq \pi_{\text{svd}}$ then set $\mathbb{Y} \leftarrow \mathbb{Y} \cup \{x^i\}$
2. Termination:
    If $|\mathbb{Y}| = n + 1$ then return $\mathbb{Y}$,
    otherwise return $\emptyset$.
```

4.3 The NM-search step

We now introduce and describe the NM-search step to insert into Algorithm 2. Given a sampling radius $\pi_{\text{radius}} \geq 1$ and a threshold parameter $\pi_{\text{svd}} > 0$, apply Algorithm 3 to attempt to build an ordered simplex $\mathbb{Y} = \{y^0, y^1, \ldots, y^n\} \subset \mathbb{R}^n$ to initiate the search. If this attempt results in $\mathbb{Y} = \emptyset$, then abort the NM-search step.

The Mads-NM algorithm will iterate and attempt to replace y^n in $\mathbb{Y}$ by generating the centroid, the reflection, the outside contraction and/or the inside contraction as in the standard NM method. However, in order to satisfy the Mads requirement that every trial point belongs to the current mesh, the following notation rounds these new tentative points to the mesh:

$x^r_\oplus$ is the reflection x^r rounded to the mesh;
$x^e_\oplus$ is the expansion x^e rounded to the mesh;
$x^{oc}_\oplus$ is the outside contraction x^{oc} rounded to the mesh;
$x^{ic}_\oplus$ is the inside contraction x^{ic} rounded to the mesh.

Notice that the centroid x^c does not need to be rounded to the mesh, because the NM algorithm never evaluates function values at the centroid.

In the unconstrained NM algorithm, Fig. 1 illustrates the four zones used to compare the trial points. An endpoint of each zone is delimited by either the best simplex vertex y^0 or by the worst one y^n. Now, in the presence of constraints, we redefine these zones so that they reflect the contribution of both the objective function f and the constraint violation function h. In order to determine at which of these trial points the function values will be evaluated, we introduce the following subsets of the simplex $\mathbb{Y}$:

$$\begin{aligned} \mathbb{Y}^0 &= \{y \in \mathbb{Y} \ : \ \nexists x \in \mathbb{Y} \text{ with } x \prec y\} \\ \mathbb{Y}^n &= \{y \in \mathbb{Y} \ : \ \nexists x \in \mathbb{Y} \text{ with } y \prec x\}. \end{aligned} \tag{3}$$

The set $\mathbb{Y}^0$ contains all vertices that are not dominated by any other vertex. The set $\mathbb{Y}^n$ contains all vertices that do not dominate any other vertex. None of these sets can be empty, and they are not necessarily disjoint. The same logic could have been applied to NM for unconstrained optimization: The point y^0 was undominated and y^n was the simplex point that did not dominate any other vertex.

The following is the counterpart of Definition 2.4, but for constrained optimization. It uses the new definition of dominance (Definition 4.1).

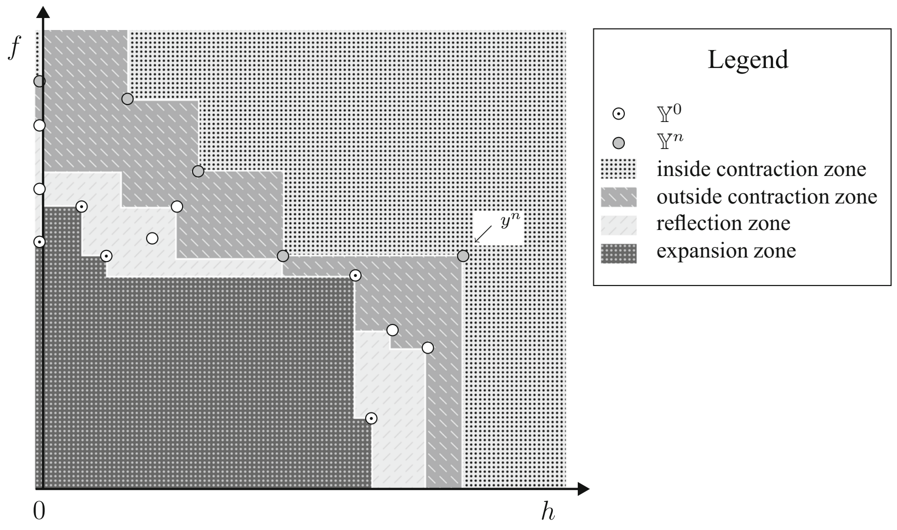


Fig. 2 Zones relative to a 16-point simplex

Definition 4.4 (**Zones for constrained optimization**) Let $\mathbb{Y} = \{y^0, y^1, \ldots, y^n\}$ be an ordered simplex in $\mathbb{R}^n$, and let $\mathbb{Y}^0$ and $\mathbb{Y}^n$ be the subsets of $\mathbb{Y}$ from Equation (3). The trial point $x \in \mathbb{R}^n$

- Belongs to the inside contraction zone if $y^i \prec x$ for some $y^i \in \mathbb{Y}^n$ or if $h(x) > h(y^n)$;
- Else, it belongs to the expansion zone if $x \prec y^i$ for some $y^i \in \mathbb{Y}^0$;
- Else, it belongs to the reflection zone if x dominates at least 2 points of $\mathbb{Y}$;
- Else, it belongs to the outside contraction zone and x dominates 0 or 1 point of $\mathbb{Y}$.

Figure 2 is the counterpart of Fig. 1 in the constrained case. Instead of only depicting the objective function on a real half-line, it takes into account the half-plane where the abscissa represents the non-negative constraint violation function h, and the ordinate is the objective function f. Figure 2 illustrates the sets $\mathbb{Y}^0$ and $\mathbb{Y}^n$ on a simplex with 16 elements. The 6 points represented by the symbol $\odot$ are the undominated vertices and the 5 ones represented by the symbol ● are the vertices that do not dominate any other one.

Algorithmic decisions are made wether the reflection point $x^r_\oplus$ dominates a point of $\mathbb{Y}^0$ (the expansion zone in Fig. 2), $x^r_\oplus$ is dominated by a point of $\mathbb{Y}^n$ (the inside contraction zone), $x^r_\oplus$ dominates 2 or more points (the reflection zone), or $x^r_\oplus$ dominates 0 or 1 points (the outside contraction zone). Recall that Definition 4.1 does not compare feasible with infeasible points, and therefore the figure is separated into two parts. The left part of the figure represents the zones for the feasible points, i.e., the points for which h equals zero. The rest of the figure concerns infeasible points.

The NM search step is compactly written as Algorithm 4.

There is a symmetry between this NM-search step and Algorithm 1, but there are three important differences. First, due to the operations that round to the mesh, it is possible that, after replacing the vertex y^n, $\mathbb{Y}$ does not form a simplex anymore. If this

Algorithm 4: The Nelder–Mead search step (NM-search)

```
Given the vertices of an initial ordered simplex $\mathbb{Y} = \{y^0, y^1, \ldots, y^n\} \subseteq V^k$.
1. Update the simplex:
    If $\mathbb{Y}$ is not a simplex, then go to 4;
    otherwise reorder $\mathbb{Y}$ and construct the sets $\mathbb{Y}^0$ and $\mathbb{Y}^n$.
2. Determine a new candidate vertex $t$:
    If $x^r_\oplus$ belongs to the inside contraction zone
        if $x^{ic}_\oplus$ belongs to the inside contraction zone, then go to 4,
        otherwise set $t = x^{ic}_\oplus$;
    else if $x^r_\oplus$ belongs to the expansion zone, then set $t = \mathsf{Best}(x^r_\oplus, x^e_\oplus)$;
    else if $x^r_\oplus$ belongs to the reflection zone, then set $t = x^r_\oplus$;
    else ($x^r_\oplus$ belongs to the outside contraction zone), set $t = \mathsf{Best}(x^r_\oplus, x^{oc}_\oplus)$.
3. Test to replace worst point:
    If $t \in V^k$ then go to 4,
    otherwise set $y^n \leftarrow t$ and go to 1.
4. Termination:
    Return the set of trial points visited during step 2.
```

is the case, then the NM-search terminates. Second, it is possible that the proposed value to replace the vertex y^n is a mesh point t that was previously visited ($t \in V^k$). In order to prevent cycling issues, the NM-search is terminated. Third, there is no need for the shrink parameter γ because the NM-search step terminates instead of shrinking the simplex. There are two reasons for not shrinking the simplex: -i- it could introduce up to n new trial points, which would increase significantly the cost of the search step; -ii- it would frequently generate points that are not on the current mesh.

When the NM-search step terminates, the Mads algorithm proceeds to the poll step as detailed in Algorithm 2. At the next Mads iteration, the NM-search step will be initiated from a new simplex determined by Algorithm 3 with the additional points generated during the poll step.

In addition to the parameters π_{radius} and π_{svd}, we introduce a third algorithmic parameter called $\pi_{\mathsf{eval}} \in \mathbb{N}$ that limits the number of function evaluations that each NM-search step can perform. In practice, this means that step 4 of the algorithm is invoked as soon as π_{eval} function evaluations are performed. The counter is reset at each new NM-search step. For readability, this parameter is not presented in Algorithm 4. In the numerical experiments that follow in the next section, the regular NM parameters are set to the values $\delta^e = 2$, $\delta^{ic} = -\frac{1}{2}$, $\delta^{oc} = \frac{1}{2}$, and the Mads algorithm with the NM search step of Algorithm 4 is denoted by Mads-NM.

5 Computational experiments

Computational experiments are conducted using the beta version 3.8.2 of the NOMAD [29] software package freely available at www.gerad.ca/nomad. All tests use the Mads strategy with $n + 1$ poll directions [6] with or without the the use of quadratic models. When both the quadratic model and NM searches are enabled, the former is performed first and the iteration opportunistically terminates in case of a success.

Data profiles [36] are presented below to assess if algorithms are successful in generating solution values close to the best objective function values. Identification of a successful run requires a convergence test. We denote x_e as the best feasible iterate obtained after e evaluations of one of the algorithm on one of the problems. The problem is said to be solved within the convergence tolerance τ when

$$\bar{f}_{\text{fea}} - f(x_e) \geq (1-\tau)\left(\bar{f}_{\text{fea}} - f^*\right)$$

where f^* is the objective function value of the best feasible points obtained by all tested algorithms on all run instances of that problem. The value of $\bar{f}_{\text{fea}}$ is a common reference for a given problem obtained by averaging the first feasible objective function values, on all run instances of that problem for all algorithms. If no feasible iterate is obtained, the convergence test is failed.

Different instances are obtained by replicating algorithm runs with different pseudo-random generator seeds. In the present work, we consider that different initial points constitute different problems.

The horizontal axis of a data profile represents the number of evaluations for problems of fixed dimension, and represents groups of $n+1$ evaluations when problems of different dimensions are involved. The vertical axis corresponds to the proportion of problems solved within a given tolerance τ. Each algorithm has its curve to allow comparison of algorithms capability to converge to the best objective function value.

The methodology of the numerical experiments is as follows.

Test on analytical problems are conducted in Sect. 5.1 to tune the algorithmic parameters. Section 5.2 compares the resulting algorithm to several well known implementations of the NM algorithm on unconstrained test problems. Finally, the algorithm is tested in Sects. 5.3, 5.4 and 5.5 on three constrained blackbox engineering test problems.

5.1 Preliminary experiments to calibrate parameters

Numerical experiments on analytical test problems are conducted to set default values to the three algorithmic parameters: The sampling radius π_{radius}, the singular value threshold π_{svd} and the evaluation budget π_{eval} of each NM call. In all figures below, the three parameters are compactly written as the triplet $\pi = (\pi_{\mathsf{radius}}, \pi_{\mathsf{svd}}, \pi_{\mathsf{eval}})$.

Mads-NM without quadratic models is tested on 87 analytical problems from the optimization literature. The characteristics and sources of theses problems are summarized in Table 1. The number of variables ranges from 2 to 20; 19 problems have constraints other than bound constraints.

A first set of runs was performed by only varying the singular value threshold $\pi_{\mathsf{svd}} \in \{0.001, 0.005, 0.01, 0.05, 0.1\}$ while fixing $\pi_{\mathsf{eval}} = 20n$ and $\pi_{\mathsf{radius}} = 4$. The parameter π_{svd} has an important influence on the performance of Mads-NM, and the best performance is obtained with $\pi_{\mathsf{svd}} = 0.01$. This value was fixed and the evaluation budget parameter was varied in a second set of runs: $\pi_{\mathsf{eval}} \in \{5n, 10n, 20n, 40n, 80n, 160n\}$. This parameter has a limited influence on the performance for values greater than $20n$ and $\pi_{\mathsf{eval}} = 80n$ is the best value. In a third set of runs, the singular

Table 1 Description of the set of 87 analytical problems

No.	Name	Source	n	m	Bnds	No.	Name	Source	n	m	Bnds
1	ARWHEAD10	[21]	10	0	No	45	PENALTY1_4	[21]	4	0	No
2	ARWHEAD20	[21]	20	0	No	46	PENALTY1_10	[21]	10	0	No
3	BARD	[35]	3	0	No	47	PENALTY1_20	[21]	20	0	No
4	BDQRTIC10	[21]	10	0	No	48	PENALTY2_4	[21]	4	0	No
5	BDQRTIC20	[21]	20	0	No	49	PENALTY2_10	[21]	10	0	No
6	BEALE	[35]	2	0	No	50	PENALTY2_20	[21]	20	0	No
7	BIGGS	[21]	6	0	No	51	PENTAGON	[30]	6	15	No
8	BOX	[35]	3	0	No	52	PIGACHE	[39]	4	11	Yes
9	BRANIN	[22]	2	0	Yes	53	POLAK2	[30]	10	0	No
10	BROWNAL5	[21]	5	0	No	54	POWELL_BS	[35]	2	0	No
11	BROWNAL7	[21]	7	0	No	55	POWELLSG4	[21]	4	0	No
12	BROWNAL10	[21]	10	0	No	56	POWELLSG8	[21]	8	0	No
13	BROWNAL20	[21]	20	0	No	57	POWELLSG12	[21]	12	0	No
14	BROWNDENNIS	[35]	4	0	No	58	POWELLSG20	[21]	20	0	No
15	BROWN_BS	[35]	2	0	No	59	RADAR	[34]	7	0	Yes
16	CHENWANG_F2	[12]	8	6	Yes	60	RANA	[25]	2	0	Yes
17	CHENWANG_F3	[12]	10	8	Yes	61	RASTRIGIN	[22]	2	0	Yes
18	CRESCENT	[3]	10	2	No	62	ROSENBROCK	[35]	2	0	Yes
19	DISK	[3]	10	1	No	63	SHOR	[30]	5	0	No
20	ELATTAR	[30]	6	0	No	64	SNAKE	[3]	2	2	No

Table 1 continued

No.	Name	Source	n	m	Bnds	No.	Name	Source	n	m	Bnds
21	EVD61	[30]	6	0	No	65	SPRING	[41]	3	4	Yes
22	FILTER	[30]	9	0	No	66	SROSENBR6	[21]	6	0	No
23	FREUDENSTEINROTH	[35]	2	0	No	67	SROSENBR8	[21]	8	0	No
24	GAUSSIAN	[35]	3	0	No	68	SROSENBR10	[21]	10	0	No
25	G210	[4]	10	2	Yes	69	SROSENBR20	[21]	20	0	No
26	G220	[4]	20	2	Yes	70	TREFETHEN	[25]	2	0	Yes
27	GRIEWANK	[22]	10	0	Yes	71	TRIDIA10	[21]	10	0	No
28	GULFRD	[35]	3	0	No	72	TRIDIA20	[21]	20	0	No
29	HELICALVALLEY	[35]	3	0	No	73	TRIGONOMETRIC	[35]	10	0	No
30	HS19	[23]	2	2	Yes	74	VARDIM8	[21]	8	0	No
31	HS78	[30]	5	0	No	75	VARDIM10	[21]	10	0	No
32	HS83	[23]	5	6	Yes	76	VARDIM20	[21]	20	0	No
33	HS114	[30]	9	6	Yes	77	WANGWANG_F3	[46]	2	0	Yes
34	JENNRICHSAMPSON	[35]	2	0	No	78	WANGWANG_F5	[46]	2	0	Yes
35	KOWALIKOSBORNE	[35]	4	0	No	79	WATSON9	[35]	9	0	No
36	MAD6	[30]	5	7	No	80	WATSON12	[35]	12	0	Yes
37	MCKINNON	[32]	2	0	No	81	WONG1	[30]	7	0	No
38	MDO	[44]	10	10	Yes	82	WONG2	[30]	10	0	No
39	MEZMONTES	[33]	2	2	Yes	83	WOODS4	[21]	4	0	No
40	MEYER	[35]	3	0	No	84	WOODS12	[21]	12	0	No
41	OPTENG_RBF	[27]	3	4	Yes	85	WOODS20	[21]	20	0	No
42	OSBORNE1	[35]	5	0	No	86	TAOWANG_F2	[43]	7	4	Yes
43	OSBORNE2	[30]	11	0	No	87	ZHAOWANG_F5	[47]	13	9	Yes
44	PBC1	[30]	5	0	No						

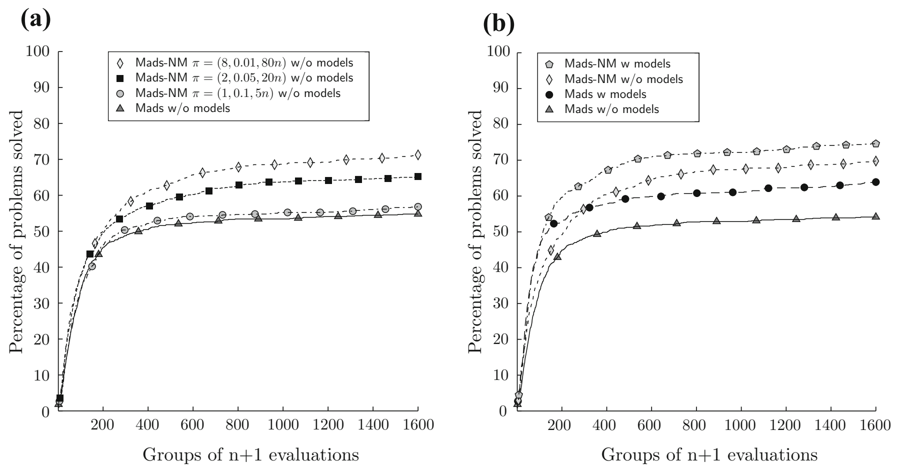


Fig. 3 Data profiles obtained with convergence tolerance $\tau = 10^{-5}$ on 10 replications of 87 analytical problems, with different NM settings (left) and with quadratic models (right). **a** Experiments with parameters π and **b** comparison with and without models

value threshold and evaluation budget parameters are fixed: $\pi_{\text{svd}} = 0.01$ and $\pi_{\text{eval}} = 80n$, and the sampling radius is varied: $\pi_{\text{radius}} \in \{1, 2, 4, 8, 16\}$. Performances are similar for $\pi_{\text{radius}} \in \{8, 16\}$. We have selected $\pi_{\text{radius}} = 8$. Finally, varying separately π_{svd} and π_{eval} shows no improvement. These intensive computational tests were possible because the analytical formulations of the 87 test problems are available.

Figure 3a shows a representative sample of data profiles obtained for three distinct sets of parameters for Mads-NM. Similar plots were obtained with $\tau = 10^{-3}$ and $\tau = 10^{-7}$ but are not presented for readability. These experiments allow us to conclude that

$$\pi = (\pi_{\text{svd}}, \pi_{\text{eval}}, \pi_{\text{radius}}) = (0.01, 80n, 8)$$

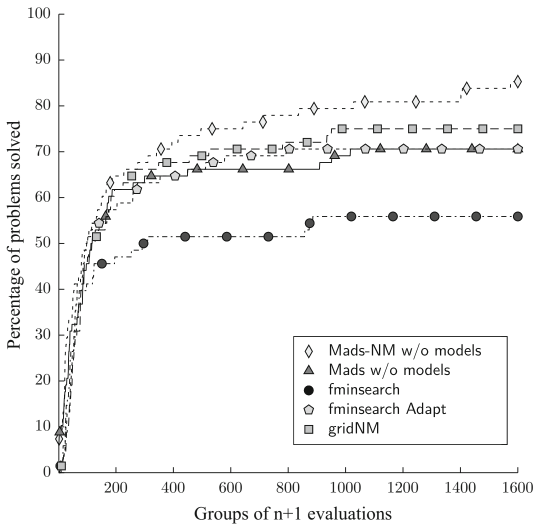
is a satisfactory set of parameters for the problems at hand, and we use these parameters for all remaining tests.

Figure 3b illustrates a second series of tests to evaluate the performance of Mads-NM with and without the use of quadratic models. The results demonstrate that Mads-NM outperforms Mads. Surprisingly Mads-NM without quadratic models performs better than Mads with quadratic models when the number of function evaluations exceeds $\sim 350 \times (n + 1)$. The combined use of quadratic models and the NM search dominates all other algorithmic variants on these analytical problems.

5.2 Comparison with NM on unconstrained optimization

The Mads-NM algorithm can be launched on unconstrained optimization problems, and may be compared to Matlab's fminsearch, an implementation [28] of the NM algorithm, fminsearch Adapt (same Matlab code as fminsearch but the expansion, contraction, and shrink parameters depend upon the dimension of the problem n as

Fig. 4 Data profiles using Mads-NM, Mads, fminsearch, fminsearch Adapt, and gridNM with a convergence tolerance of $\tau = 10^{-5}$ on one replication of 68 test problems without constraints other than bounds



suggested in [20]) and gridNM, an implementation [10] of the grid restrained NM algorithm. In order to make this comparison, we extract from the previous 87 analytical problems, the ones without constraints other than bounds ($m = 0$ in Table 1). This yields a subset of 68 test problems.

With fminsearch, a single initial point is considered from which an initial simplex is automatically created and the algorithm does not depend on a random number generator. The same initial simplex selection as in fminsearch is used for the gridNM and fminsearch Adapt algorithms. For fair comparison, both the Mads and Mads-NM algorithms are run only once for all test problems.

The data profiles in Fig. 4 illustrate that gridNM, fminsearch Adapt and both Mads and Mads-NM without quadratic models outperform Matlab's fminsearch. Again Mads-NM is the dominating algorithm.

5.3 The LOCKWOOD problems

The Mads and Mads-NM algorithms with quadratic models are tested to solve the basic version of a pump-and-treat groundwater remediation problem from Montana Lockwood Solvent Groundwater Plume Site [31]. Each initial point defines a LOCKWOOD problem. Solving the problem consists in minimizing the operating costs subject to 2 constraints on the flux of two contaminant plumes obtained from the Bluebird simulator [16]. The problem has 6 design variables bounded in $[0; 20000]^6$.

Figure 5 shows the optimization history when solving 20 LOCKWOOD problems from different initial points randomly selected in the hyper-rectangle $[0; 20000]^6$. The graphs plot the evolution of the incumbent solution value versus the number of function evaluations. The right-hand-side plot zooms in on lower values of the objective function values. For 5 initial points, the Mads-NM algorithm fails to reach the feasible region and Mads fails 4 times. However, the figure shows that Mads-NM finds more

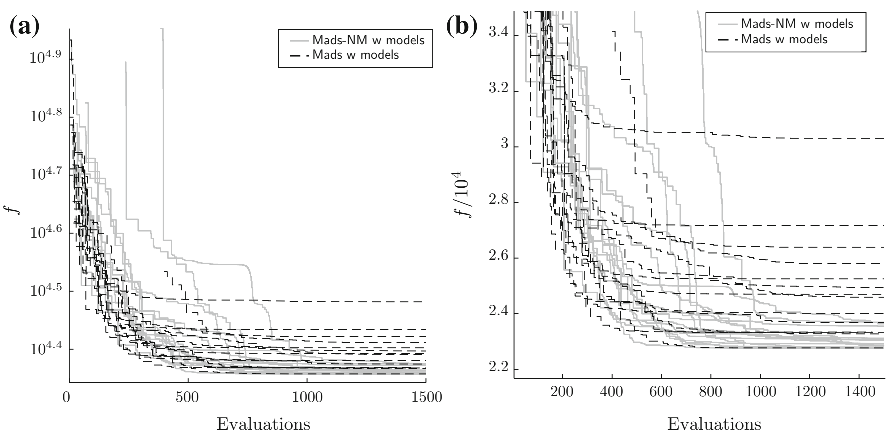


Fig. 5 Optimization history on 20 LOCKWOOD problems (right plot is a zoom on low objective values)

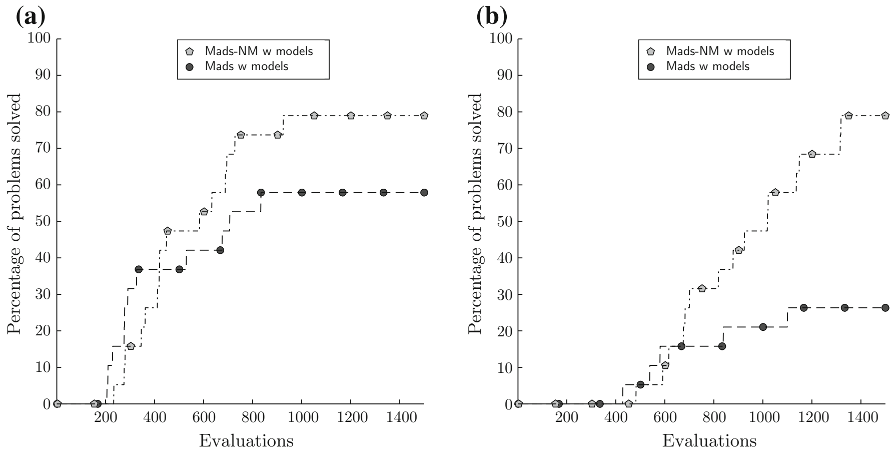


Fig. 6 Data profiles obtained with convergence tolerance τ on 20 LOCKWOOD problems. **a** $\tau = 10^{-1}$ and **b** $\tau = 10^{-2}$

solutions with a lower objective function value than Mads. This observation is also clearly apparent in the data profiles of Fig. 6.

5.4 The MDO problems

The Mads and Mads-NM algorithms with quadratic models are tested to solve a simple multidisciplinary wing design optimization problem [44]. Each initial point defines a MDO problem. Solving the problem consists in maximizing the range of an aircraft subject to 10 constraints. The problem has 10 scaled design variables bounded in the hyper-rectangle $[0; 100]^{10}$.

Figure 7a shows the optimization history when solving 20 MDO problems on different initial points using 1500 function evaluations or less. The initial points are integers

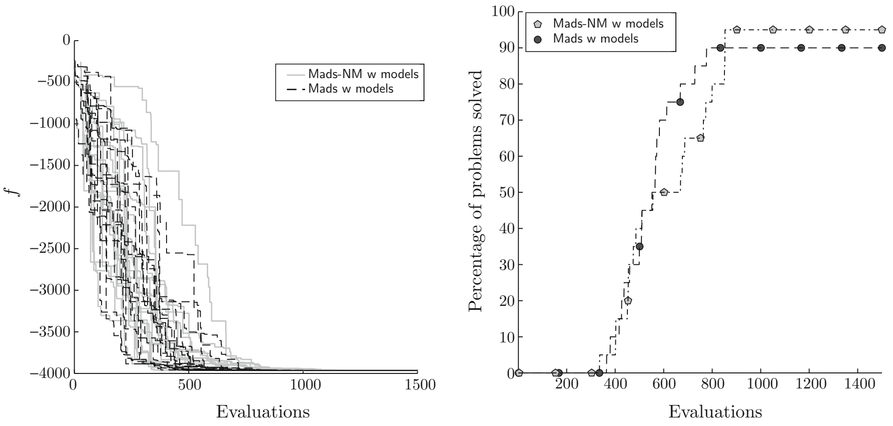


Fig. 7 Mads and Mads-NM with quadratic models on 20 MDO problems. **a** Optimization history and **b** Data profiles with $\tau = 10^{-2}$

randomly selected within the bounds. For 1 initial point, the Mads-NM algorithm fails to reach the feasible region and Mads fails 2 times. The plot of the optimization history and the data profiles from Fig. 7 show that the behaviour of both the Mads-NM and Mads algorithm is similar. For large number of evaluations, Mads-NM solves slightly more problems than Mads.

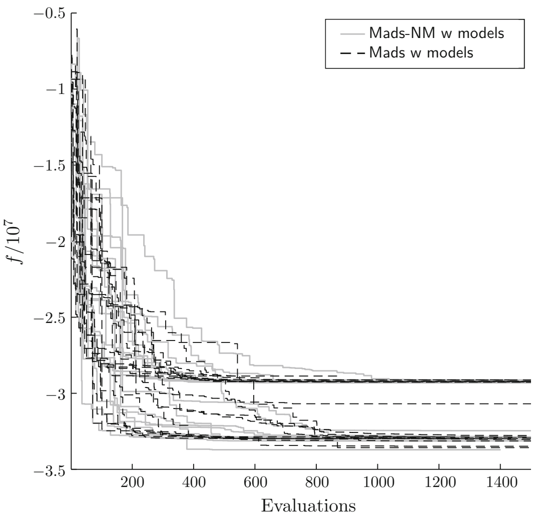
In contrast with the previous subsections, the numerical experiments on the MDO problems do not reveal a dominant algorithm.

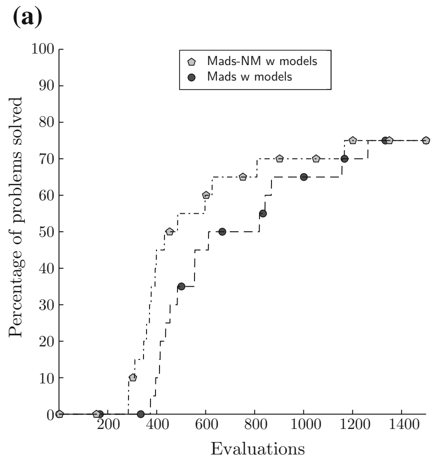
5.5 The STYRENE problems

The Mads and Mads-NM algorithms with quadratic models are tested to optimize a styrene production process [1]. The simulation of the chemical process relies on a series of interdependent calculation blocks using common numerical methods such as Runge-Kutta, Newton, fixed point and also chemical related solvers. Once the simulation of the process is successfully completed, the constraints and objective can be evaluated during a post-processing. The objective is to maximize the net present value of the styrene production process with 9 industrial and environmental regulations constraints. The simulation and post-processing are combined in a blackbox. If the simulation of the chemical process cannot succeed for an iterate, a blackbox evaluation failure is returned.

In this work, a STYRENE problem possesses eight 8 independent variables influencing the styrene production process. The variables considered during optimization are all scaled and bounded in $X = [0; 100]^8$. As a reference, we use the initial point x^{ref} provided in [1]. From this reference point, 20 STYRENE problems have been created by randomly generating integer initial point in an hyper-rectangle of radius 10 centred at x^{ref} (the points are projected on X if necessary). For all initial points, both the Mads and Mads-NM algorithms reach the feasible region from every initial point. The history plots of the Mads and Mads-NM runs using 1500 evaluations or

Fig. 8 Optimization history on 20 STYRENE problems





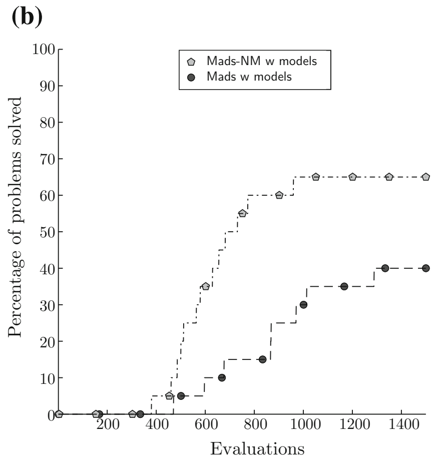


Fig. 9 Data profiles obtained with two convergence tolerances on 20 STYRENE problems. **a** $\tau = 10^{-2}$ and **b** $\tau = 10^{-3}$

less are presented in Fig. 8. Different local minimizers are reached by the algorithms, but the figure does not clearly indicate which algorithm is preferable.

Data profiles with convergence tolerances of $\tau = 10^{-2}$ and 10^{-3} are presented in Fig. 9. The profiles with $\tau = 10^{-3}$ shows a dominance of Mads-NM over the Mads algorithm.

6 Discussion

The paper introduces a way to extend the NM direct search algorithm so that it handles general inequality constraints. This is achieved by defining a NM search step within the Mads algorithm. The NM search points form simplices, ordered by a transitive relation involving both the objective and constraint violation function values. The simplices are reflected, expanded and contracted as in the original algorithm, except that the trial points are rounded to the current mesh. This last operation is necessary in order to inherit from the rich Mads hierarchical convergence analysis. Each NM search step is interrupted as soon as the vertices fail to form a simplex, or when a vertex is replaced by a previously visited point. The resulting algorithm is applicable to both unconstrained and inequality constrained blackbox optimization problems.

Numerical experiments on unconstrained optimization problems show that the resulting Mads-NM algorithm outperforms the Matlab implementations of NM (fminsearch and fminsearch Adapt), the gridNM algorithm, as well as the previous implementation of Mads with and without the use of quadratic models. Experiments on three constrained engineering problems suggest that the behaviour of Mads-NM and Mads is comparable during the first few hundreds of function evaluations. But as the number of function evaluations grows, the Mads-NM algorithm typically finds better feasible solutions than Mads. The difference in the percentage of problems solved in the data profiles is approximately 5% for the MDO problems (with $\tau = 10^{-2}$), 25% for the STYRENE problems (with $\tau = 10^{-3}$) and more than 50% for the LOCKWOOD problems (with $\tau = 10^{-2}$).

The beta version 3.8.2 of the NOMAD blackbox optimization software uses the NM search step as described in the present paper, and contains all test problems considered with their reference initial point.

Acknowledgements This work is supported by NSERC CRD Grant (#RDCPJ 490744 - 15) in collaboration with Hydro-Québec and Rio Tinto. The authors wish to thank Shawn Mattot, Genetha Gray, and Stefan Wild for making the LOCKWOOD problem available.

References

1. Audet, C., Béchard, V., Le Digabel, S.: Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. J. Global Optim. **41**(2), 299–318 (2008)
2. Audet, C., Dennis Jr., J.E.: Mesh adaptive direct search algorithms for constrained optimization. SIAM J. Optim. **17**(1), 188–217 (2006)
3. Audet, C., Dennis Jr., J.E.: A progressive barrier for derivative-free nonlinear programming. SIAM J. Optim. **20**(1), 445–472 (2009)
4. Audet, C., Dennis Jr., J.E., Le Digabel, S.: Parallel space decomposition of the mesh adaptive direct search algorithm. SIAM J. Optim. **19**(3), 1150–1170 (2008)
5. Audet, C., Hare, W.: Derivative-Free and Blackbox Optimization. Springer Series in Operations Research and Financial Engineering. Springer, Berlin (2017)
6. Audet, C., Ianni, A., Le Digabel, S., Tribes, C.: Reducing the number of function evaluations in mesh adaptive direct search algorithms. SIAM J. Optim. **24**(2), 621–642 (2014)
7. Audet, C., Kokkolaras, M., Le Digabel, S., Talgorn, B.: Order-based error for managing ensembles of surrogates in derivative-free optimization. J. Global Optim. **70**(3), 645–675 (2018)
8. Audet, C., Le Digabel, S., Tribes, C.: Dynamic scaling in the mesh adaptive direct search algorithm for blackbox optimization. Optim. Eng. **17**(2), 333–358 (2016)
9. Brea, E.: An extension of Nelder–Mead method to nonlinear mixed-integer optimization problems. Rev. Int. Métod. Numér. Cálc Diseño Ing. **29**(3), 163–174 (2013)
10. Bűrmen, Á., Puhan, J., Tuma, T.: Grid restrained Nelder–Mead algorithm. Comput. Optim. Appl. **34**(3), 359–375 (2006)

11. Chang, K.H.: Stochastic Nelder–Mead simplex method—a new globally convergent direct search method for simulation optimization. Eur. J. Oper. Res. **220**(3), 684–694 (2012)
12. Chen, X., Wang, N.: Optimization of short-time gasoline blending scheduling problem with a DNA based hybrid genetic algorithm. Chem. Eng. Process. **49**(10), 1076–1083 (2010)
13. Clarke, F.H.: Optimization and Nonsmooth Analysis. Wiley, New York (1983). (reissued in 1990 by SIAM Publications, Philadelphia, as Vol. 5 in the series Classics in Applied Mathematics)
14. Conn, A.R., Le Digabel, S.: Use of quadratic models with mesh-adaptive direct search for constrained black box optimization. Optim. Methods Softw. **28**(1), 139–158 (2013)
15. Coope, I.D., Price, C.J.: Frame-based methods for unconstrained optimization. J. Optim. Theory Appl. **107**(2), 261–274 (2000)
16. Craig, J.: Bluebird developer manual. http://www.civil.uwaterloo.ca/jrcraig/pdf/bluebird_developers_manual.pdf (2002). Accessed 11 June 2018
17. Dennis Jr., J.E., Woods, D.J.: Optimization on microcomputers: The Nelder–Mead simplex algorithm. In: Wouk, A. (ed.) New Computing Environments: Microcomputers in Large-Scale Computing, pp. 116–122. Society for Industrial and Applied Mathematics, Philadelphia, PA (1987)
18. Fletcher, R., Leyffer, S.: Nonlinear programming without a penalty function. Math. Program. Ser. A **91**, 239–269 (2002)
19. Fletcher, R., Leyffer, S., Toint, PhL: On the global convergence of a filter-SQP algorithm. SIAM J. Optim. **13**(1), 44–59 (2002)
20. Gao, F., Han, L.: Implementing the Nelder–Mead simplex algorithm with adaptive parameters. Comput. Optim. Appl. **51**(1), 259–277 (2012)
21. Gould, N.I.M., Orban, D., Toint, PhL: CUTEr (and SifDec): a constrained and unconstrained testing environment, revisited. ACM Trans. Math. Softw. **29**(4), 373–394 (2003)
22. Hedar, A.-R.: Global optimization test problems. http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm (2017). Accessed 11 June 2018
23. Hock, W., Schittkowski, K.: Test Examples for Nonlinear Programming Codes, Lecture Notes in Economics and Mathematical Systems, vol. 187. Springer, Berlin (1981)
24. Jahn, J.: Introduction to the Theory of Nonlinear Optimization. Springer, Berlin (1994)
25. Jamil, M., Yang, X.-S.: A literature survey of benchmark functions for global optimisation problems. Int. J. Math. Modell. Numer. Optim. **4**(2), 150–194 (2013)
26. Kelley, C.T.: Detection and remediation of stagnation in the Nelder–Mead algorithm using a sufficient decrease condition. SIAM J. Optim. **10**(1), 43–55 (1999)
27. Kitayama, S., Arakawa, M., Yamazaki, K.: Sequential approximate optimization using radial basis function network for engineering optimization. Optim. Eng. **12**(4), 535–557 (2011)
28. Lagarias, J.C., Reeds, J.A., Wright, M.H., Wright, P.E.: Convergence properties of the Nelder–Mead simplex method in low dimensions. SIAM J. Optim. **9**, 112–147 (1998)
29. Le Digabel, S.: Algorithm 909: NOMAD: nonlinear optimization with the MADS algorithm. ACM Trans. Math. Softw. **37**(4), 44:1–44:15 (2011)
30. Lukšan, L., Vlček, J.: Test problems for nonsmooth unconstrained and linearly constrained optimization. Technical Report V-798, ICS AS CR (2000)
31. Matott, L.S., Leung, K., Sim, J.: Application of MATLAB and Python optimizers to two case studies involving groundwater flow and contaminant transport modeling. Comput. Geosci. **37**(11), 1894–1899 (2011)
32. McKinnon, K.I.M.: Convergence of the Nelder–Mead simplex method to a nonstationary point. SIAM J. Optim. **9**(1), 148–158 (1998)
33. Mezura-Montes, E., Coello, C.A.: Useful infeasible solutions in engineering optimization with evolutionary algorithms. In: Proceedings of the 4th Mexican International Conference on Advances in Artificial Intelligence, MICAI'05, pp. 652–662, Springer, Berlin (2005)
34. Mladenović, N., Petrović, J., Kovačević-Vujčić, V., Čangalović, M.: Solving spread spectrum radar polyphase code design problem by tabu search and variable neighbourhood search. Eur. J. Oper. Res. **151**(2), 389–399 (2003)
35. Moré, J.J., Garbow, B.S., Hillstrom, Kenneth E.: Testing unconstrained optimization software. ACM Trans. Math. Softw. **7**(1), 17–41 (1981)
36. Moré, J.J., Wild, S.M.: Benchmarking derivative-free optimization algorithms. SIAM J. Optim. **20**(1), 172–191 (2009)
37. Nazareth, L., Tseng, P.: Gilding the lily: a variant of the Nelder–Mead algorithm based on golden-section search. Comput. Optim. Appl. **22**, 133–144 (2002)

38. Nelder, J.A., Mead, R.: A simplex method for function minimization. Comput. J. **7**(4), 308–313 (1965)
39. Pigache, F., Messine, F., Nogarede, B.: Optimal design of piezoelectric transformers: a rational approach based on an analytical model and a deterministic global optimization. IEEE Trans. Ultrason. Ferroelectr. Freq. Control **54**(7), 1293–1302 (2007)
40. Price, C.J., Coope, I.D., Byatt, D.: A convergent variant of the Nelder–Mead algorithm. J. Optim. Theory Appl. **113**(1), 5–19 (2002)
41. Rodríguez, J.F., Renaud, J.E., Watson, L.T.: Trust region augmented Lagrangian methods for sequential response surface approximation and optimization. J. Mech. Des. **120**(1), 58–66 (1998). 03
42. Strasser, M.: Übertrangung des Optimierungsverfahrens von Nelder und Mead auf restringierte Probleme. Diploma thesis, Numerical Mathematics Group, Technical University of Darmstadt, Germany (1994)
43. Tao, J., Wang, N.: DNA double helix based hybrid GA for the gasoline blending recipe optimization problem. Chem. Eng. Technol. **31**(3), 440–451 (2008)
44. Tribes, C., Dubé, J.-F., Trépanier, J.-Y.: Decomposition of multidisciplinary optimization problems: formulations and application to a simplified wing design. Eng. Optim. **37**(8), 775–796 (2005)
45. Tseng, P.: Fortified-descent simplicial search method: a general approach. SIAM J. Optim. **10**(1), 269–288 (1999)
46. Wang, K., Wang, N.: A novel RNA genetic algorithm for parameter estimation of dynamic systems. Chem. Eng. Res. Des. **88**(11), 1485–1493 (2010)
47. Zhao, J., Wang, N.: A bio-inspired algorithm based on membrane computing and its application to gasoline blending scheduling. Comput. Chem. Eng. **35**(2), 272–283 (2011)